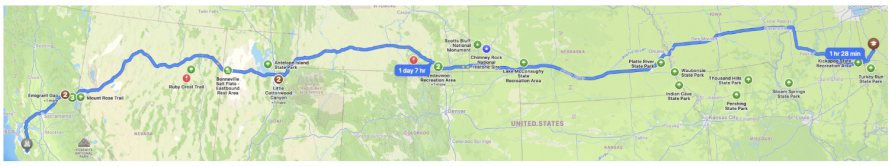
Exercise 5.10. Imagine you are planning a long road trip to San Jose, CA and you've already fixed your route (through Illinois, Iowa, Nebraska, Wyoming, Utah, Nevada, and then California via Tahoe).



You don't mind the drive, but you hate having to stop to get gas. What a pain! And some stations take longer than others. The car can drive at most 500 miles before needing to stop to get gas. The high-level goal is to plan your gas stops to minimize the amount of time spent pumping gas, while making sure you don't run out of gas between them.

Design and analyze an algorithm to compute the minimum amount of time one has to spend at gas stations in going from location 0 to location Z, while ensuring that you never drive more than 500 miles without visiting a gas station. (You should return $+\infty$ if it is impossible to do so.)

Guiding Questions

Our problem is to design an algorithm, that given distances $D[1, n]$, times $T[1, n]$ and destination Z, compute the minimum time to get to Z, stopping for gas every 500 miles or less.

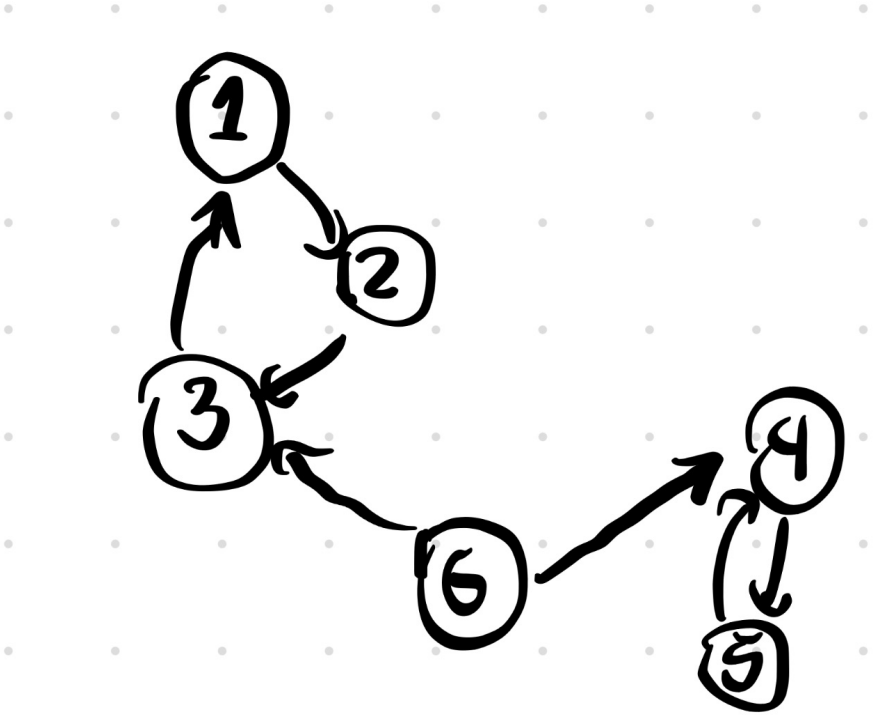
1. Write out your recursive spec. What would be good to keep track of?

   Let DP(_____) =the minimum time it takes to get to _____,
   given that we do not go over 500 miles per station.

2. *Recursive spec implementation.* This is clearly a minimization problem, so we just need to fill in the blanks:

$$\texttt{DP}(\_\_\_) = \min_{i \in [n] \text{ such that } \_\_\_\_\_ < 500} (\texttt{DP}(\_\_\_) + \_\_\_\_\_) .$$

3. How do we use this to solve the original problem?



Problem (Warmup Questions/Test Your Understanding).

Searching Graphs

1. Count the number of strongly connected components in the following graph.
2. In the same graph, find the topological sort of the condensed graph (where each scc is a single vertex). How many topo sorts exist?
3. (True/False) In a graph $G = (V, E)$, all vertices reachable from vertex $v \in V$ are marked by `DFS`(v).
4. (True/False) In the post DFS-ordering, the first vertex is always the sink.

Shortest Paths

1. (True/False) Dijsktra's algorithm is a *single-source* shortest paths algorthm. That is, Dijkstra only finds the shortest path of a single source.
2. (True/False) Dijkstra with a priority queue takes $O(m \log n)$ time

Exercise 6.4. Let $G = (V, E)$. Design and analyze an algorithm for each of the following problems.

1. Decide if there is a vertex x such that x can reach all other vertices.
2. Decide if there is a vertex x such that x can be reached by all other vertices.
3. Decide if there is a vertex x such that every vertex can either reach x or be reached from x.

Guiding Questions

1. How do you check reachability of a single vertex x? That is, how do you compute the vertices v such that there is a path from x to v? Note that part 1 asks us to decide if such a vertex x exists, so we do not know which vertex x before hand.
2. Recall one can visit all vertices using `DFS-Driver` from the lecture notes (i.e., for each unmarked vertex v remaining, run `DFS(v)`). What can you say about the last unmarked vertex that `DFS-Driver` runs `DFS` on?
3. Hint for part 2 : *change the graph, not the algorithm.*
4. Hint for part 3 : cycles can be annoying when trying to solve this problem. How can we 'condense/contract' them?

Exercise 7.1. Each of the following problems describes a graph problem over an unweighted directed graph $G = (V, E)$ with m edges and n vertices. Each problem can be solved by constructing an auxiliary graph and calling a shortest path algorithm from this chapter as a black box. For each problem, (a) design an auxiliary graph and shortest path problem, (b) indicate which algorithm to apply and how, and (c) analyze the running time. No proof of correctness is required.

1. For two vertices s and t, compute the length of the shortest walk from s to t with an even number of edges.[3]
2. For two vertices s and t, compute the length of the shortest walk from s to t where the number of edges is a multiple of 5.

Hint: Don't change the algorithm. Change the graph. I drew a few graphs for you to play around with.